Knowledge and Skill Representations for Robotized Production ⋆

Anders Björkelund, Jacek Malec, Klas Nilsson, Pierre Nugues

Department of Computer Science, Lund University, Sweden, e-mail:
{*anders.bjorkelund, jacek.malec, klas.nilsson, pierre.nugues*}*@ cs.lth.se*

Abstract:
Model-based systems in control are a means to utilize efficiently human knowledge and achieve high performance. While models consisting of formalized knowledge are used during the engineering step, running systems usually do not contain a high-level, symbolic representation of the control and most of its properties, typically named numerical parameters. On a system level and beyond the plant data, there is also a need to represent the meaning of the data such that deployment and fault analysis could be augmented with partly automated inference based on the semantics of the data. To that end, we extended the formalized knowledge traditionally used in control to include the control purpose, engineering assumption, quality, involved state machines, and so on. We then represented the control semantics in a format that allows an easier extraction of information using querying and reasoning. It aims at making knowledge in control engineering reusable so that it can be shipped together with the control systems. We implemented prototypes that include automatic conversion of plant data from AutomationML into RDF triples, as well as the automated extraction of control properties, the conversion of parameters, and their storage in the same triple store. Although these techniques are standard within the semantic web community, we believe that our robotic prototypes for semantic control represent a novel approach.

Keywords: Model-based control, Knowledge representation, System architectures, Autonomous control, Industrial robots

1. INTRODUCTION

Model-based systems in control are a means to utilize efficiently human knowledge and achieve high performance. Models are formalized knowledge, but that formalization is usually limited to the mathematics used during engineering of the control; a running control system does not contain a high-level representation of the control and most of its properties. Properties include the normal control parameters that are typically represented as named numerical entities, which are used in user interfaces for tuning and deployment. However, the meaning of the control parameters is normally only documented in a human-readable form. That is, even if mathematical descriptions are part of a documentation, there is no machine-understandable information structure that supports storage and reuse of engineering knowledge, which in turn would be useful for improved, possibly semi-automatic, engineering tools.

The control parameters, and a few other items such as log data for monitoring, already exist as part of controller interfaces today, and are suitable for initial prototyping. To store and reuse the engineering knowledge, we need to extend the formalized knowledge to include also the purpose of control, the engineering assumption, the quality of control as a function of the provided resources, state machines, and so on. In this paper, we describe a method to represent explicitly the semantics of the control so that querying and reasoning is made possible.

⋆ The research leading to these results has received funding from the European Union's seventh framework program (FP7/2007-2013) under grant agreement N^{O} 230902.

2. CONTROL KNOWLEDGE

Accumulating control knowledge may be approached from different directions. Starting from actual controllers is a bottom-up approach. On a system level, there are corresponding needs that form a top-down approach starting from the overall plant or manufacturing line. We need then to represent not only the plant data, but also its meaning such that deployment and fault analysis can be partly automated using the semantic descriptions. The top-down and bottom-up approaches need to be coherent to facilitate the desired reuse of control engineering knowledge and it should be possible, to some extent, to ship this knowledge embedded with the control systems for later on-site usage.

While it is easy to pinpoint the above mentioned goals, it is hard to show how that could and should be accomplished. The normal system engineering approach would be to include it all in a list of requirements, use standardized interfaces, encode possible variation in flexible data structures, and develop the complete system. One can even claim that some of the existing systems already embody what we suggest. However, as we experienced in both academia and industry, control system solutions result from human innovations in a manner that is too expensive to track (extensive re-engineering of tools and interfaces) with the established system engineering approaches. In practice, even a well-designed (but pre-engineered) system gets too restrictive with respect to unforeseen upcoming changes. Therefore we need to introduce techniques for storing and managing some of the human knowledge within our domain.

3. SCOPE

The domain selected for this paper is robotics in manufacturing, which sets severe requirements on flexibility and performance. Flexibility includes interaction with human operators for tuning the production and for handling unforeseen error situations. Performance, despite hard-to-predict variations of workpieces and in process dynamics, includes the use of application-specific sensing and feedback control (not pre-engineered with the robot, and subject to change at the production site). Thus, the bottom-up focus is on productive sensor-based robot motions, which we refer to as skills in the following. The top-down scenario is the manufacturing line with much equipment already having their properties modeled, although not in a manner that extends well when new features are added. A coherent approach that extends efficiently (with no or little reprogramming when another XML schema or another type of feedback control is added) is the ultimate challenge for this work.

Research on knowledge representation for robotics has provided a multitude of alternatives ranging from application data being stored in individual files, to database-driven platforms supporting information types covering large facilities. Although such systems are often presented as open and generic, most of them are indeed closed systems, with proprietary know-how and specialized tools necessary to create, maintain, and use these knowledge bases. Therefore, as a basis for our contribution we propose and prototype an architectural framework based on *semantic web* techniques (Antoniou and van Harlmelen, 2008) allowing a standardized representation, storage, and distribution of knowledge. This architecture offers the potential of making automation data modular and extensible, while well-defined interfaces and APIs provide possibility of accessing it as needed.

The paper is divided as follows. We first discuss related and previous work. Next, we present a device and skill ontology that we chose as the starting point for this investigation. Then we discuss how this ontology can be incorporated in a *knowledge integration framework* (KIF). Finally, we report the results of first practical experiments. The paper ends with conclusions and plans for future work.

4. PREVIOUS WORK

The semantic web has been designed to represent concepts, objects, and their relationships. It should enable systems to (Buitelaar, 2007):

- Encode and interpret data using a rich hierarchical and relational structure;
- Extract data and integrate them into applications;
- Share data with a common format.

Semantic approaches have previously been applied to automation and robotics systems. The DEPUIS (Design of environmental-friendly products using information standards) (Amato et al., 2008) and S-TEN (Intelligent Self-describing Technical and Environmental Networks) [1] are examples that investigated the compatibility of the STEP standard, ISO 10303, with the *Web ontology language* (OWL) and their mutual conversion. However, both lines of research seem to have been discontinued.

Earlier work on skill-based inspection and assembly for reconfigurable automation systems (Malec et al., 2007) described the possibility to base reconfiguration reasoning on common semantical grounds using a manufacturing ontology. Although it did not use the linked data paradigm (Berners-Lee, 2006), it showed the advantages of having a common semantics for all used data. However, common semantics should not be confused with standardized semantics, otherwise this would assume a-priori knowledge about the *unforeseen* new knowledge. This issue is addressed by the *open-world assumption* imposed on the representation.

There have been several attempts to codify production knowledge in the form of suitable ontologies and associated tools. Lastra and Delamer (2008) provided an overview of this expanding field. Kim et al. (2006) described an early attempt focusing on collaboration issues in the design process. Naumann et al. (2010) is a recent report of activities complementing our work.

Knowledge in robotics systems covers a considerable set of disciplines and categories: logic information, finite–state machines and discrete-event systems, differential-algebraic systems, geometric and kinematics models, databases and first-order logic, and robot task programs to name a few. All these categories use different data representations and even for the same category, the formats of engineering tools may also be different. This lack of standardization is a problem to manufacturers, as they cannot easily switch equipment, and causes well-known difficulties to customize the production.

AutomationML [2] is a standardized markup language that attempts to model and unify all kinds of information used by engineering tools. It covers plant topology, geometry and kinematics, logic information, reference and relations, and referencing other formats (Drath, 2010). The upper-level part of AutomationML uses the CAEX data exchange format. CAEX is a semantic middleware framework to store hierarchical object information, properties, and libraries (Fedai et al., 2003). It represents topology information in the form of plants, cells, components, attributes, interfaces, relations, and references (AutomationML, 2009). This CAEX top-level connects the different data formats used downstream by the different categories of engineering tools; for example COLLADA (Barnes and Finch, 2008) for geometry and kinematics data and PLCopen (PLCopen, 2003) for logic data.

Persson et al. (2010) is a recent attempt to exploit AutomationML data for creating semantic knowledge base for automatic production. We develop this concept further in this paper.

5. THE KNOWLEDGE INTEGRATION FRAMEWORK

We designed an architecture to represent, store, adapt, and distribute knowledge used in robotized production, where we abstract the components as data sources. We call it the *knowledge integration framework* (KIF). In our robotic applications, we assume that data is normally available in the AutomationML exchange format (Drath, 2010).

The semantic approach enables the different sources of models to share a common, standardized storage and exchange format and makes it easier to implement inference procedures about those models. Choosing one common representation (RDF)

[1] http://www.s-ten.eu/

[2] http://www.automationml.org/

with well-defined semantics is a clear advantage over proprietary exchange formats or even commonly used standardized XML.

The choice of RDF solves only part of the problem, namely representation of discrete data, expressible as logical relations between objects. Other kinds of models (procedural, equational, hybrid) will require extensions of the models we currently express in RDF. One example is geometry, which according to AutomationML should be expressed in COLLADA.

To cope with unforeseen changes that might invalidate earlier assumptions on the control, it is central to adopt the open-world assumption (OWA), that Allemang and Hendler (2008, page 11) formulated as:

> An open world [...] is one in which we must assume at any time that new information could come to light, and we may draw no conclusions that rely on assuming that the information available at any one point is all the information available.

The required mindset to develop such knowledge-based systems is quite different from that of current control engineering and software engineering approaches. This is however manageable by focusing on the knowledge as such.

6. SKILL AND DEVICE REPRESENTATION

Malec et al. (2007) presented an initial attempt towards a manufacturing ontology, which was centered around the concepts of devices and their capabilities that we refer to as skills. Other types of basic concepts and related (nonprocedural) knowledge included tasks, workpieces, and the notion of an environment. Most of these concepts can be specified on at least two levels:

- abstract descriptions, like a *pickup* skill, and
- instantiated, concrete ones, like the gripper G_1 gripping windshield W_{23} in factory F_7.

Skill models involve explicit logical conditions and implicit behavioral models referenced in the ontology.

The *Device Library* is an important part of the manufacturing ontology. It contains the device descriptions that show as elements (leaves) in the ontology. It is designed so that a reasoning system can extract compatibility constraints, functional information, and parameters The device library consists of *virtual parts*, plugged in as needed and as available. It could be maintained by device manufacturers.

6.1 Contents of the Manufacturing Ontology

The original ontology contained knowledge about abstract skills and devices. Figure 1 shows a top-level view of it. This is an area where a categorization is easy to carry out as, for example, with the concept of *vacuum gripper*, which is a subconcept of *gripper*, which in turn is a subconcept of *device*. Similarly, a *vacuum-pickup* skill is a subconcept of *pickup* skill. In this work, we extended the ontology with process knowledge complementing the so-called production triangle: process, product, and resource.

To have a more complete graph, we also represented semantic connections between the skills and the devices. In particular, each device is able to perform one or more skills, and each skill can be performed by one or more devices. The hierarchical

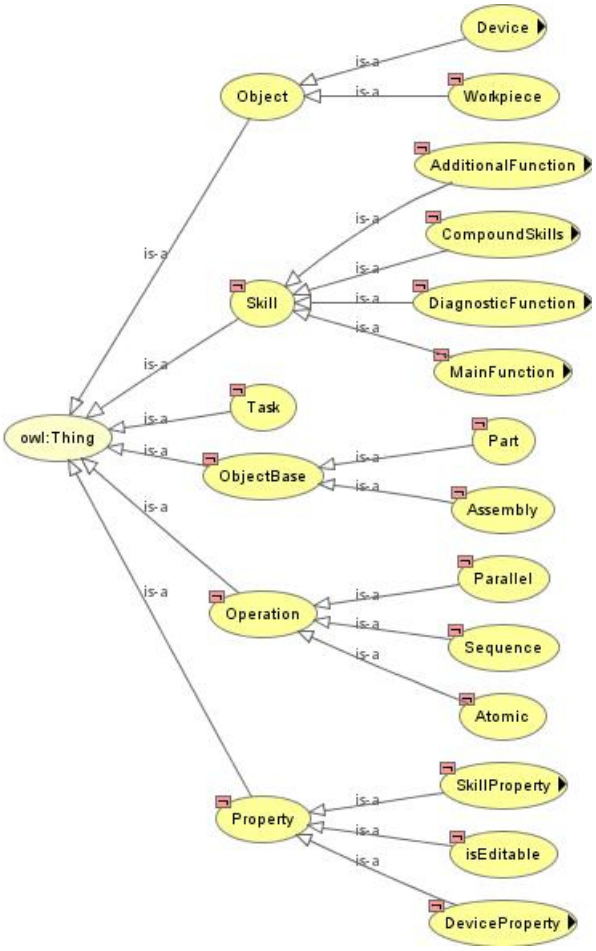

Fig. 1. The ontology top level.

structure of the ontology allows, for instance, to specify that a *vacuum gripper* has a *vacuum-grasp* skill, from which it may be automatically deduced that it is a *gripper* and has, therefore, a more generic *grasp* skill as well.

However, when addressing robots, it might be necessary to specify that a *robot* uses a tool, e.g. a *gripper* to perform some *skill*. Doing it properly is outside the representational power of an ontology, at least in the form it has now. We use the concept of *compound devices* for this purpose. See Malec et al. (2007) for details.

The tasks constitute means of achieving a particular manufacturing goal and a reasoner needs to be presented with at least some description of this goal. It needs to be able to reason about rationale for each task and about reasons why a particular device and particular parameters were chosen. At the very least, it needs a list of criteria which distinguishes acceptable execution of a task from unacceptable ones, considered as errors.

6.2 Device Library

The part of the ontology containing information about the physical devices and their properties is called the *Device Library*. Although virtually indistinguishable from the rest of the ontology, it forms a conceptually separate system with respect to the rest of the ontology. The distributed nature of the KIF plays an important role in allowing this to happen seamlessly.

One important aspect of the device library is that it must contain not only static information about the devices like its name, properties expressed using discrete parameters, e.g., weight, power consumption, or accuracy, but also geometric information, which may be parametrized in some way (for this purpose we assume a COLLADA specification pointed to by an appropriate URI) and some description of the dynamic behavior of the device. The behavior may be described in many ways and it will be the visualization/simulation or even deployment interface that will determine what kind of descriptions will be necessary or useful for its needs. However, there may, and usually will, be more than one behavioral model, extending the utility of information contained in the device library.

6.3 Development of the Ontology

The original ontology focused on the connection of skills and devices. The relation to product and process aspects was only contextual and provided implicitly in the reasoning algorithm. The revised ontology presented here enhances the definition of *skill* and stresses the interaction with all three nodes of the production triangle: product, process, and resource. Namely, the skills also include the coordination of actions

The extensions introduced for the KIF consist mainly of:

- The knowledge representation is based on persistent triple stores (not restricted by the tables of classical databases);
- Reasoners are not committed to yet, and no particular architectural solution for this task is chosen. Therefore, the generic picture of a knowledge-based system, with pluggable external reasoners is perfectly valid, while the specific, blackboard-based utility-function mechanism of reasoning in Malec et al. (2007) has not been considered as relevant for KIF.

Summarizing this section: KIF exploits the previously developed skill and device ontology in a creative manner, augmenting it with other facets and enabling other kinds of reasoning than those which were originally possible. The next section will present some details of the solutions adopted and will illustrate the approach.

7. EXPOSING THE SEMANTICS

7.1 The Resource Description Framework

The resource description framework (RDF) (RDF, 2004) is a initiative of the World Wide Web consortium (W3C) to bring semantics to the web. The framework represents information as collections of triples consisting of a subject, a predicate, also called a property, and an object.

A collection of triples forms a directed graph, where the predicates correspond to the arc labels and the subjects and objects to the pairs of connected vertices. Subjects and predicates are unambiguously named using uniform resource identifiers (URIs); objects are either resources or literal values, i.e. numbers, strings, dates, etc. Literal values can only correspond to terminal nodes of a graph. The line

```
<rosetta#d1e7> caex-xml:hasName "Cam_3".
```

is an example of a triple that connects the subject node `<rosetta#d1e7>` to a literal string, `"Cam_3"`. The predicate gives a meaning to this relation: the object is a name

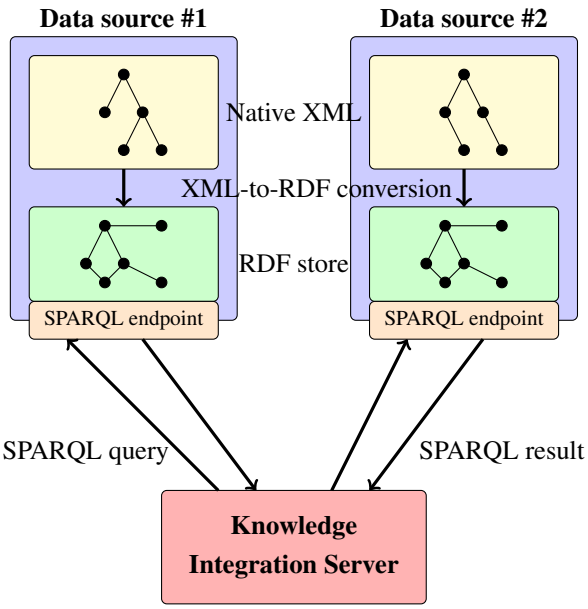

Fig. 2. The communication mechanism.

of the subject. The prefix `caex-xml` is the abbreviation of a complete URI and e.g., in our current implementation, it stands for

```
http://asimov.ludat.lth.se/2009/09/caex-xml.owl#
```

The RDF model is close to concepts used in classical logic (De Bruijn et al., 2005). In the traditional predicate logic notation, each RDF triple `(Subject Predicate Object)` would correspond to the statement: `Predicate(Subject, Object)`. In addition to standardization, the possibility to reformulate the RDF model into a logic setting enables its users to benefit from a considerable amount of results and tools.

7.2 Exposing AutomationML as RDF Triples

To expose the semantics of CAEX and PLCopen files and make their information content explicit, we convert them to RDF triples. Following the DBpedia method (Auer et al., 2007), we implemented a procedure that transforms the data sources used in an AutomationML environment into RDF repositories and we make them accessible using the SPARQL query protocol. The procedure comprises the following steps:

(1) Extract and transform data from all the knowledge sources into RDF triples;
(2) Expose the resulting graphs and make them accessible using RDF repositories. For some nodes, we used the Linked Data method to associate the node URIs to HTTP accessible data;
(3) Access and modify the graphs from a central integration server using a SPARQL update endpoint or another update mechanism.

The system overall architecture thus enables the interconnection of distributed data sources in a transparent manner. Each of these sources can utilize existing XML-based data, which is very common in industry today. Figure 2 shows the communication between the data sources and the knowledge integration server.

In order to convert an XML document, we use the *document object model* (DOM) of the original XML documents and

we apply a syntactic mapping to produce the RDF triples, as described in Persson et al. (2010). The conversion procedure uses rules that take the elements of the source DOM tree (i.e. `xsd:elements` and `xsd:attributes`) and create the corresponding nodes of the output RDF graph. The RDF predicates are automatically generated from the XML element names by the concatenation of the prefix `has` and the element name.

The procedure is implemented in the XSLT language, where XSLT builds a DOM representation of the XML input document, traverses the tree, and applies the rules to produce the resulting RDF graph. The transformations are then straightforward and the graph can easily be understood by the means of the XML schema describing the original document. Overall, this procedure transforms XML documents – including the AutomationML standard suite – into queryable data sources. A drawback is that we lose the ability to validate the consistency of the RDF graph with respect to the underlying XML schema. This could be solved by constructing an appropriate OWL ontology and using a reasoner.

7.3 Hosting and accessing RDF Triples

We host the converted triples in a network-enabled RDF repository using Sesame server developed by the OpenRDF[3] community. It is deployed as Java servlets with a web interface so that users can easily save, export, and inspect graphs. In addition to the subject–predicate–object triple, Sesame associates a context to each statement. A repository can thus contain one or more named graphs.

Once the data sources are available in the RDF repository, one can use SPARQL (SPARQL, 2008) to express queries over the graphs. SPARQL extracts patterns from them in a way similar to Prolog or Datalog (Schenk, 2007). SPARQL can deliver the output as a table (using the `SELECT` keyword) or format it as an RDF graph (using `CONSTRUCT`). RDF access is thus possible with Sesame via a SPARQL endpoint, programmatically via the platform API, or directly using Sesame's web interface.

7.4 The prototype

The KIF server is the center point of an architecture consisting in particular of an engineering station, a high-level controller and a low-level firmware/hardware interface, among other units.

An engineering station is used to enable a user to interact with the system: select or modify the task to be performed, select devices to be used and the physical environment of the work cell. Although the station possesses simulation capabilities, the generic knowledge about devices and tasks, stored in an appropriate set of ontologies, is retrieved from KIF, using the methods described above. When the work cell and task are specified, they are exported in AutomationML-based format to the KIF.

The high-level controller carries out the task synthesis from the abstract, device-independent form stored in KIF to the concrete one, containing the code to be executed in the robot and work cell controller. At all levels of abstractions, we represent the tasks as transition systems of various kinds: depending on the software used at particular level it might be a PLCopen-like structure, a sequential function chart of IEC 61131-3, a state chart in the Simulink environment, with robot-specific actions typically implemented in plain robot programming languages of the native system, such as RAPID or KRL depending on the target robot.

The synthesis is facilitated by the knowledge available in KIF. The first phase, device-independent, is focused on task decomposition leading to a task specification in which every step is realizable by the available devices. As the robot prototype is to assemble small parts, possibly with two hands, the reasoning involved deals with fixtures, grasping, mounting, setting, moving, etc. The second phase instantiates the generic task specification with parameters and details specific to the concrete devices used in assembly. For this purpose, the KIF contains a library of available devices and possible ways of performing assembly, together with the corresponding controller code.

The KIF is also used to store data gathered during run-time, which is later exploited for learning and optimization. The optimization may be local and applicable only to the current set of devices and a concrete task, or may be global, spanning over a number of realizations using replaceable devices, or even several tasks.

The tests performed currently within the framework of EU FP7 Rosetta project are done using the Robot Studio software environment from ABB as the engineering station, our custom KIF server implemented on top of Sesame, and a custom high-level controller of the Rosetta project. The robots we target include various manipulators from Kuka and ABB, including the recent FRIDA two-handed manipulator.

8. CONCLUSIONS

The work presented in this paper aims at creating a knowledge repository to support model-based engineering practices, in particular in the context of robotized industry. The repository would support knowledge reuse and learning, making it easier to adapt the robots or, more generally, the available resources to new tasks when a production line is changed. We have designed and are currently further prototyping a knowledge integration framework. First experiments show that we can handle large amounts of data in the form of AutomationML documents describing complete plants as well as hardware-close information about the control used in a particular process. One of the crucial steps is the use of an underlying ontology that allows us to handle models used in control and automation in an efficient fashion.

Targeting future usage in real production environments, the ongoing efforts include enhancements influencing how the data and control knowledge are gathered and represented. We investigate the replacement of the current syntax-based translation of the XML formats used in AutomationML and PLCopen, which may not be well suited for semantic processing or reasoning as their structure is far from a natural rule formalism. Interfaces towards engineering tools are currently being developed, so that the contents of KIF may be on one hand properly visualized, and on the other hand exploited by the control system developers. The KIF has also been attached to the run-time control system of an industrial robot, which will also be used for gathering on-line knowledge while the production cell is being run.

[3] `http://www.openrdf.org/`

To make more sense of the graphs gathered from different sources and describing different kind of models, one or more additional ontologies could be created that would still fit the initial RDF graph. This is the approach used by Runde et al. (2009) on a limited set of examples. As multiple inheritance is allowed in RDF, this does not contradict the ontology derived from the XML schema, but could be considered as an additional layer on top of the initial graph.

REFERENCES

Allemang, D. and Hendler, J. (2008). *Semantic Web for the Working Ontologist; Effective Modeling in RDFS and OWL*. Morgan Kaufmann.

Amato, A., Moreno, A., and Swindells, N. (2008). Depuis project: Design of environmentally-friendly products using information standards. In G. Cascini (ed.), *Computer-Aided Innovation (CAI)*, volume 277 of *IFIP International Federation for Information Processing*, 135–143. Springer Boston.

Antoniou, G. and van Harlmelen, F. (2008). *A semantic web primer*. The MIT Press, second edition.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In Aberer (ed.), *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*. Busan, Korea.

AutomationML (2009). AutomationML specification. Part 1 – Architecture and general requirements. Technical report, AutomationML Group.

Barnes, M. and Finch, E.L. (2008). COLLADA – digital asset schema release 1.5.0. Technical report, Khronos Group.

Berners-Lee, T. (2006). Linked data. `http://www.w3.org/DesignIssues/LinkedData.html`.

Buitelaar, P. (2007). On the role of natural language processing in a data-driven approach to the ontology life-cycle. Keynote talk at TALN, Toulouse, France (http://olp.dfki.de/ontoselect/).

De Bruijn, J., Franconi, E., and Tessaris, S. (2005). Logical reconstruction of normative RDF. In *OWL: Experiences and Directions Workshop*.

Drath, R. (ed.) (2010). *Datenaustausch in der Anlagenplanung mit AutomationML. Integration von CAEX, PLCopen, XML und COLLADA*. Springer.

Fedai, M., Epple, U., Drath, R., and Fay, A. (2003). A metamodel for generic data exchange between various CAE systems. In I. Troch and F. Breitenecker (eds.), *Proceedings of 4th Mathmod Conference*, 1247–1256. Vienna.

Kim, K.Y., Manley, D.G., and Yang, H. (2006). Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design*, 38, 1233–1250.

Lastra, J.L.M. and Delamer, I.M. (2008). Automation 2.0: Current trends in factory automation. In *6th IEEE International Conference on Industrial Informatics*. IEEE.

Malec, J., Nilsson, A., Nilsson, K., and Nowaczyk, S. (2007). Knowledge-based reconfiguration of automation systems. In *Proc. of the IEEE Conference on Automation Science and Engineering*, 170–175. IEEE.

Naumann, M., Bengel, M., and Verl, A. (2010). Automatic generation of robot applications using a knowledge integration framework. In *Proc. International Symposium on Robotics ISR 2010*. Munich, Germany.

Persson, J., Gallois, A., Björkelund, A., Hafdell, L., Haage, M., Malec, J., Nilsson, K., and Nugues, P. (2010). A knowledge integration framework for robotics. In *Proceedings of the joint conference of the 41st International Symposium on Robotics (ISR 2010) and the 6th German Conference on Robotics (ROBOTIK 2010)*, 1068–1075.

PLCopen (2003). IEC 61131-3: Programmable controllers – part 3: Programming languages. Technical report, International Electrotechnical Commission.

RDF (2004). RDF/XML syntax specification. `http://www.w3.org/TR/rdf-syntax-grammar/`.

Runde, S., Güttel, K., and Fay, A. (2009). Transformation von CAEX-Anlagenplanungsdaten in OWL. Eine Anwendung von Technologien des Semantic Web in der Automatisierungstechnik. In *AUTOMATION 2009. Der Automationskongress in Deutschland*, 175–178. VDI Verlag.

Schenk, S. (2007). A SPARQL semantics based on Datalog. In *KI 2007: Advances in Artificial Intelligence*, 160–174. Springer.

SPARQL (2008). SPARQL protocol and RDF query language. `http://www.w3.org/TR/rdf-sparql-query/`.